SERBIAN JOURNAL OF ELECTRICAL ENGINEERING
Vol. 21, No. 1, February 2024, 53-71
UDC: 004.76:004.4`416]:620.9 **DOI: https://doi.org/10.2298/SJEE2401053S**

Intelligent and Metaheuristic Task Scheduling for Cloud using Black Widow Optimization Algorithm

Sadhana Selvakumar[1], Pandiarajan Subramanian[1]

Abstract: Cloud computing is an internet-based infrastructure for services such as computations, storage, etc., hosted on physical machines. The machines on cloud infrastructure scales between a few tens to thousands of machines that are linked in an unstructured way. In cloud computing, minimizing energy consumption and its associated costs is the primary goal while preserving efficiency and performance. It progresses the system's overall productivity, reliability, and availability. Furthermore, reducing energy use not only lowers energy expenses but also helps to safeguard our natural environment by lowering carbon emissions. The objective of our proposed work is to reduce energy usage in the cloud environment and enhance its performance. We propose a hybrid approach that incorporates an energy-aware self-governing task scheduler, namely, Artificial Neural Network (ANN), and a metaheuristic Black Widow Optimization (BWO) algorithm to solve the optimization issues. Our suggested task scheduler focuses on minimizing energy consumption, improving the makespan, and reducing the operating cost while keeping a low number of active cloud racks. The cloud environment is highly scalable in this scenario since we adopt a metaheuristic BWO algorithm. CloudSim simulation framework is used for implementation and experimental analysis.

Keywords: Black Widow Optimization, Energy efficiency, Bio-inspired, Artificial Neural Networks, Cloud computing.

1 Introduction

The high-level massive applications in cloud computing have recently been successful in addressing the expansion of computational needs for enormous application scales since the cloud provides flexible and durable computing resources available for usage in the pay-per-use model [1]. Numerous tasks are usually performed "as a service" model on the cloud environment. Software-as-a-Service (SaaS) collaboration paradigm provides users with cloud applications over the Internet, which are accessed on a desktop or workstation using dependent customer applications, for instance, web browser. It is typically used for cloud-

[1]Department of Computer Science and Engineering, KIT-KalaignarKarunanidhi Institute of Technology, Coimbatore, India; E-mails: sadhana.dsk@gmail.com; evergreenpandi@gmail.com

based service applications such as webmail, social networking, video-sharing, and text document editing [2].

Platform-as-a-Service (PaaS) offers a suitable environment for creating applications to upgrade, test, and accommodate their applications. Moreover, it offers to the distribution of large-scale applications to scalable and elastic computing systems. End users can search for the apt use by virtualized computer equipment known as Virtual Machines (VM) with an advanced CPU processor, storage zone, memory, and bandwidth, in accordance with Infrastructure-as-a-Service (IaaS). Numerous VM scenarios are offered to consumers at different prices, which assist users in managing computer resources at the end of the day. IaaS offers end-users three key benefits. The primary benefit is that consumers use cloud resources or services based on their requirements and employ a pay-per-use model for infrastructure, like how essential services, such as electricity, gas, and water, might be paid. Based on the needs of its application, users can utilize this to contract or raise their resource agreement. The second advantage of IaaS cloud computing is the direct supply of resources that enhances user application production. The third benefit is that users can acquire rental resources at any time and from any location, based on the desired service level. The requirement of resources is still a well-known problem in conducting large-scale jobs on the IaaS cloud [3].

As demand for high-performance computer resources increases, the Cloud Data Center's (CDC) energy usage rate also increases [4]. The energy required by computing resources and the corresponding cooling system contributes to high energy and carbon emissions. The data centers use roughly 1.4% of the world's electricity usage, which grows at a rate of 12.5% annually [5]. Power consumption by processing units, cooling facilities and storage facilities, is 42.0%, 15.4%, and 19.3% respectively, and is predicted to increase 12% per annum. It generates up to 2% of the world's carbon emissions and is forecast to increase rapidly [6]. In addition, electricity expenditures represent about 15% of data center costs [7]. It is a prominent problem to overcome in terms of energy efficiency. The energy-saving hardware technologies and efficient CDC management activities can reduce electricity-based costs and carbon emissions [8]. Poor management of computing resources can lead to excessive energy consumption and a shorter lifespan.

It is essential to look at the challenges of task planning to reduce energy usage while enhancing performance (Makespan). There are challenges found in the task scheduling strategies and resource allocation in the cloud. Metaheuristic optimization algorithms are employed for solving task scheduling problems in the cloud. Metaheuristic methods have led to favourable outcomes in the search region by limiting the search solution area in terms of efficient performance [9]. Several research utilized bio-inspired optimization algorithms [10 – 13] such as

Genetic Algorithm (GA), Hybrid Particle Swarm Optimization (HPSO), Intelligent Water Drops (IWD), and Cuckoo Search Algorithm. However, these meta-heuristic approaches might be still problematic: 1) tendency to decrease overall performance as the complexity of problems increases; 2) more time consumed on the same problem instance; 3) the status of the cloud environment is not taken into consideration, unable to cope with the dynamics in the cloud environment. Moreover, these techniques might be affected by early convergence (being stuck in local search), an uneven balance of searching strategies between local and global, and their exploitation is not as stable as their search capability. Those concerns are addressed using an optimal Artificial Neural Network [14].

In this study, we suggested an independent task scheduler using ANN to lessen energy and execution overhead to improve system performance. The randomly created independent tasks are scheduled in a heterogeneous environment. The rest of the research work is structured as follows. The related works based on the usage of optimization algorithms in cloud task scheduling are reviewed in Section 2. The cloud environment system model and problem formulation are explained in Sections 3 and 4, respectively. Section 5 presents the proposed ANN-BWO algorithm and Section 6 discusses the experiment setup, the results, and their performance analysis. Finally, the paper ends with the conclusion of the work.

2 Literature Review

Data centers enormous power, and several researchers have recently worked on cloud task scheduling problems with different optimization targets such as cost saving, implementation time, dependability, consumption of energy, and so on [9 – 11]. Algorithms such as MaxMin, MinMin, and Optimized makespan can produce solutions with low overhead performance but are not sufficiently efficient [15]. Meta-heuristic algorithms also build a cost and time efficient schedule and provide sub-optimal results.

In addition, many earlier studies apply Machine Learning (ML) algorithms [16 – 17] to resolve numerous challenges in resource management, such as ANN [14], and Deep Reinforcement Learning (DRL). The DRL automates the process of establishing optimum VM settings depending on the present application requirements. Researchers likewise focus on the automation of the resource assignment procedure on application demand scales. The applications are not allocated with the existing computing resources by RL. Instead, they have dynamically scaled the previously allocated resources, like the processor capability and memory, according to the application size.

A new method based on the modified PSO algorithm is suggested in [18] to address task scheduling problems, local optimum and premature convergence problems. The results demonstrated that the proposed modified PSO can reduce

the total cost compared to other suggested methods. Garg and Goraya [19] proposed an energy-efficient heuristic algorithm for task scheduling in a virtualized cloud with a deadline constraint. This approach comprises two instances to enhance energy efficiency by executing the maximum workload of the VM and saving the most energy in the VM's idle state. Yassa et al. [20] resolved the problem of scheduling tasks using a variety of computing schemes. They presented a novel multi-objective optimization method for cloud problems by introducing the hybrid PSO algorithm for determining the performance of optimal scheduling.

Safari and Khorsand [21] offered a novel technique for minimizing energy usage in the job distribution of multiple VMs on heterogeneous cloud systems. To preserve service quality, they integrated a power-aware and list-based scheduling technique with a novel method under time constraints. As a result, the purpose of task scheduling is to enhance performance and energy usage. Similarly, the authors in [22] researched energy-efficient task scheduling in a Cloud Data Center environment. Aside from DVFS energy management techniques and VM reusage, the authors in [21] developed and implemented the EATS-FFD algorithm, serving as the primary task-scheduling strategy. The goal is to reduce energy consumption while ensuring QoS.

The research works in [23−25] used ML strategies for estimating the usage of resources, with a metaheuristic approach, which is consistent for predictions and scheduling jobs. However, an energy-efficient scheduling decision directly utilizes a recursive neural network model by them. The authors in [26] presented a directed search optimization intending to minimize makespan without taking high energy usage. They iteratively trained an ANN, like any other metaheuristic method with faster convergence. Yang et al. [27] have utilized ML techniques to improve the efficiency of large-scale systems. It tries to improve the Quality of Service (QoS), handle cloud resource requests, and consolidate workloads.

3 System Model

A modern CDC usually includes a big room with multiple rows of racks and numerous computer resources and supporting appliances. Each rack has a Cluster-Level switch connected to the other racks, and each computer inside it has the next set of computer resources connected inside the rack. The other Information Technology (IT) equipment, including air conditioners, power distribution components, temperature sensors, and air handling units assists the primary computing resources. Fig. 1 displays the current CDC, showing how the computing resources of physical hosts may be grouped into rack unities together and how each computing resource can represent a Virtual Machine (VM), which assigns and executes real jobs.

Many users submit several separate tasks, which are then arranged for the amount of VM resources available. We may observe that job assignments for certain VM sets schedule tasks automatically through diverse racks. Rack's conscious planning adds further drive for energy efficiency and the use of resources. Rack-conscious scheduling decides the apt rack for a specific activity or collection of tasks. It can be an active and lightly loaded rack instead of turning on a completely fresh one because the rack and associated gear, like a fan set, air blower, interconnection bays, etc., save a considerable computing and cooling energy. Once the rack is selected for specific tasks, the tasks are scheduled using the task scheduler. This scheduling method spreads the scheduler over the whole data center.

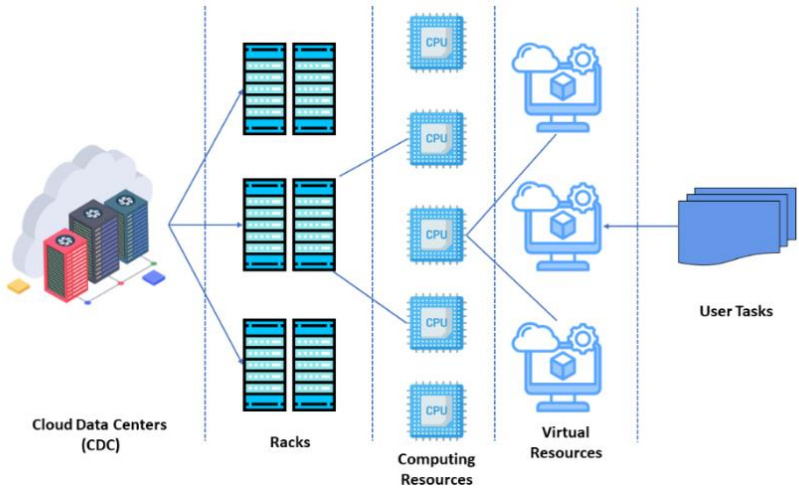

Fig. 1 – *System Model for the Proposed Work.*

4 Problem Formulation

In this section, the problem formulation for the energy consumption and time estimation for execution are described.

4.1 Estimation of execution time

One of our goals is to enhance the performance, i.e., the time required to accomplish certain activities using computer resources. To estimate the time required for a virtual computing resource $V_0^{m,n}$ to execute k^{th} task on n^{th} physical machine of m^{th} rack and calculate it as (1):

$$ET_k^{m,n,o} = \frac{T_k}{C_o^{m,n}}, \tag{1}$$

where, T_k is the working load of the k^{th} task and $C_o^{m,n}$ refers to the capacity of virtual computing resource. We presume that each virtual computer resource works on the host computing resource space shared. So, we are summing up the time to do tasks allocated to certain virtual computer resources, which are defined in (2) to calculate the time to finish a virtual computer resource:

$$FT_o^{m,n} = \sum_{\forall k \in \left| TS_o^{m,n} \right|} ET_k^{m,n,o}, \tag{2}$$

where $FT_o^{m,n}$ represents the Virtual Computing Resource and $TS_o^{m,n}$ is the Virtual Computing Resource Task Set. The completion time of the high-performance server machine or multiple computer resource VM (P_n^m) for all planned activities is determined as stated in (3): As virtual computing resources function independently,

$$FT_n^m = \max_{\forall o \in P_n^m} FT_o^{m,n}, \tag{3}$$

where FT_n^m denotes a completed time of the P_n^m resource for computing. Likewise, a rack takes time to complete each scheduled work, as set out in (4): The maximum time among the computing resources for completion:

$$FT_m = \max_{\forall n \in R_m} FT_n^m, \tag{4}$$

where FT_m is the maximum time taken by m^{th} rack to do the allocated tasks.

4.2 Energy consumption

Our subsequent goal is to lower the energy consumption in data centers while preserving performance. As a result, to carry out specific tasks, we must estimate the energy consumed using computer resources. Energy consumption in cloud data centers is controlled primarily by resources such as electricity consumed by the processing units, storage, and cooling. We use an energy consumption based on linear regression on the links between CPU utilization and energy consumption usage that works well even when Dynamic Voltage and Frequency Scaling (DVFS) is used to simulate energy consumption in the cloud. DVFS is a technique that adapts input voltage and frequency to present working loads, as its name suggests. The function of CPU use is electricity consumption. Using this info, we define $P_t^{m,n}$ energy consumption as follows:

$$EC_n^m = P_t^{m,n} \times ET_m, \tag{5}$$

where the EC_n^m table of electricity for the execution of a job during the execution period is the P_n^m calculating resource ET_m, and $P_t^{m,n}$ is the current time energy spent by the CPU for the use of the P_n^m computing resource.

$$EC_m = \sum_{\forall n \in R_m} EC_n^m \ . \tag{6}$$

Using the above information, we can compute the total energy consumed by a data center as follows:

$$EC = \sum_{\forall m \in CDC} EC_m \ . \tag{7}$$

In this study, we have taken the challenge of job planning in a cloud setting to reduce energy consumption and increase application performance. We aim to lower the overhead performance of the algorithm at the same time. We plan individual tasks on a set of computer resources that are accessible and arranged together into a rack unit. Each computing resource in the CDC can have many virtual resources to perform tasks autonomously. Each of them has millions of instructions that can be expressed as a workload. In addition, processing capability is known in advance together with the energy ingesting rate of each computing resource. At the cluster and rack levels, every new task queue is large enough to accommodate all incoming jobs.

5 Proposed Methodology

In this section, we propose a hybrid approach by combining the ability of the ANN scheduler, which is energy-efficient [14] and the metaheuristic optimization algorithm called BWO [28] within the CDC to decide on the scheduling agreement to the present cloud environment.

5.1 Artificial neural network scheduler

In [14], ANN and its applications were briefly reviewed by the authors. ANN is enthused by human brain functioning and the way information is processed. The brain consists of thousands of cells known as neurons, formed with the help of axons, are connected again with many cells. External stimuli are taken as input by dendrites and generate electrical impulses through the brain network. Each neuron determines whether or not to transmit the message to another network neuron. Likewise, ANN is a human brain digital model. It has several nodes in the human brain akin to actual neurons. These (neurons) nodes are linked to and can interact with each other via communication linkages (dendrites, axons). A node collects the input data and then executes straightforward operations of the data. Each communiqué link is weighted, and they determine which communications to send and which not. Fig. 2 is a rudimentary digital neural network with a two-node input, a four-node hidden, and a two-node output (of two neurons).

By adjusting these weights, ANN can learn sophisticated correlations of effects in the external environment. In our situation, ANN's external environment is the task burden and cloud environment from which ANN gets input (X) and creates bidding details and scheduling choices as outputs via ANN. The ANN learning process is commonly known as the training process. Two forms of

training are available, 1) The input data is supervised along with the wanted outputs, which are fed into ANN, and 2) the input is not supervised. The data is sent to ANN with a certain power function. The supervised training method is usually utilized by ANN, making ANN excellent for tasks like prediction and classification. Therefore, we adopt a scaled gradient back propagation approach with tan and sígma to train the ANN that the research academy generally uses for supervised training.

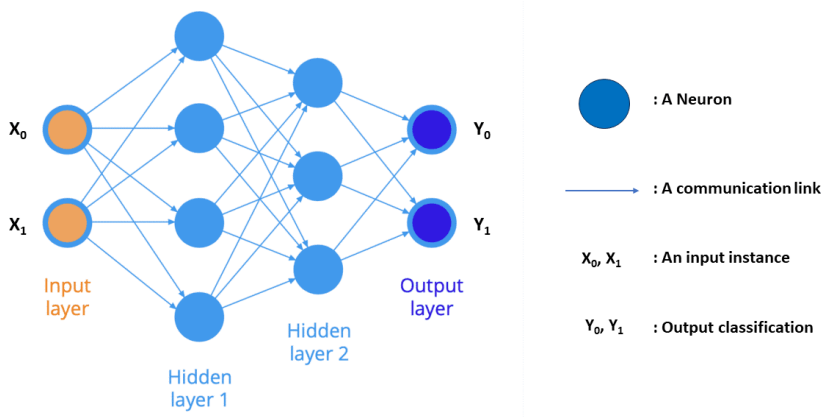

Fig. 2 – *Simple Architecture of ANN.*

The basic principle of back-propagation is to use one learning input signal to change weights on the link in the direction of lesser errors using the scaled conjugate gradient algorithm. The suggested ANN-based task scheduler can accurately learn the complicated task load correlations and the instantaneous status of the cloud environment. It significantly minimizes the overhead execution because it does not work iteratively. It is adaptive in nature by considering the present state/dynamics of the cloud as an input to scheduling decisions. Once an ANN-based task scheduler has been trained by sufficient data (experience) instances, it can forecast the expected outputs of instances, perhaps not experienced. Several training instances have been established so that a neural network can understand the complexity associated with these instances. Thus, a huge network of 5 cached layers is explored, each layer with 20 nodes (total of 100 cached layer nodes), including an input and output layer.

5.2 Black widow optimization algorithm [28–29]

In general, ANN cannot adjust the amount of input and output nodes after the training because a new neuron with input or output nodes must be introduced. However, it needs new training with changes in the number of inputs or output nodes. Researchers employ ANN with a high level of input that takes future requirements into account to handle the challenge of scalability. But in our

scenario, we cannot take the same strategy as the cloud environment that quickly grows over time, which is dynamic in nature. To solve this issue, we use the bioinspired BWO algorithm in our proposed ANN classifier, which is described as follows. Black widow spiders have unique mating behaviour, including a stage called cannibalism. In this stage, spiders with unsuitable fitness are ignored from the circle, leading to rapid convergence and resulting in preventing local optima. It is realized that BWO can obtain the best global solution by maintaining a balance between exploration and exploitation. Hence, the BWO algorithm is considered for this research to solve optimization problems in task scheduling. The following subsections discuss the different stages of the BWO algorithm.

a) Initial population

The values of the difficulty must be defined as an acceptable construction to solve the current problem to solve an optimization problem. This construction is known as "chromosome" or "particle position" in black widow meta-heuristic optimization language, and "widow" in the NBW method. Black Widow Spiders are included in the algorithm as potential solutions to any problem. Variable problems are represented by individual black widow spiders. A test function array is what this structure is in this article. An initial spider population is used to produce an entrant widow matrix of size N pop N var, and then perform the optimization process. To begin the mating process, parent couples are randomly selected.

b) Procreate

A new generation is born as the sets of each other mate. Each pair mates individually from the rest of their network simultaneously. Each time they mated, they lay roughly 1,000 eggs, but only a few offspring make it. To duplicate the random numbers in the widow's matrix, an Alpha matrix must be produced in this algorithm. In the following equation (8), x 1 and x 2 are parents' offspring and are used to construct them.

$$\begin{cases} y_1 = a \times x_1 + (1-a) \times x_2, \\ y_2 = (1-a) \times x_1 + a \times x_2. \end{cases} \tag{8}$$

There is no need to duplicate randomly selected integers because this procedure is repeated N var/2 times. Ultimately, children and mothers join the ranks and are ranked by their fitness scores. Some of the best people have been admitted to the newly formed population because of their cannibal rating. All couples are affected by these rules.

c) Cannibalism

There are three distinct sorts of male predators represented in this picture. To begin with, a black widow is eating her spouse during or after sexual relations. We could distinguish women and men based on their fitness levels using

algorithms. Strong spiders eat their pathetic allies in the second type of cannibalism. Cannibalism Score (CR) determines the number of survivors in these algorithms. A third type of ogre is observed occasionally, which feeds on its mother in the form of little spiders. To distinguish between weak and powerful spiders, we employ fitness values.

d) Mutation

Randomly chosen Mute Pop members from the population are selected at this time. Two fundamentals that can be changed in the array are illustrated in Fig.3 for each solution. Based on the mutation rates, the mute population is designed.

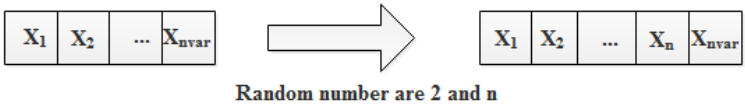

Fig. 3 – *Mutation in BWO.*

e) Convergence

There are three possible outcomes for this algorithm's conclusion: (a) a predetermined number of repeats; (b) For the vast majority of delegates, the best compliance grade for widows remains unchanged; (c) Definition of precision. As illustrated in the picture, the key phases of NBW are depicted. Some of the challenges with optimizing the NBW test utilization are discussed in the next section. A certain level of precision is considered while evaluating an experimental algorithm because the best solutions for testing activities are already well-known. As a result, Section 4 defines the maximum number of repetitions as stop conditions for the trials.

f) Parameter setting

To get the most out of the proposed plan, several conditions must be met. It includes the rate of purchases (PP), the rate of consumption (CR), and the rate of conversion (PM). The settings need to be tweaked for the algorithm to locate the optimal answers. To optimize the proposed plan, conditions such as enhancing parameter management, improving navigation on any local User Management (UM) platform, and improving global location capabilities must be met. Since the precise number of parameters can control the balance between the absorption phase and inspection phase, it is possible. PP, CR, and PM all play a role in the proposed algorithm's control scheme. The ownership ratio (PP) governs how many people are involved in a company and its products. This parameter provides a wide range of options for defining a study location because it regulates the development of distinct offspring. Unqualified people are excluded from the CR cannibal operator’s population control parameter. The Prime Minister represents

the proportion of voters who have shifted their allegiance. These parameters can be used to achieve a balance between the use and the search of these parameters. In the local phase, agents can be controlled by this factor, leading to better resolution.

The purpose of the ANN scheduler is to program incoming tasks in the rack with a view to improve performance and reduce energy usage. The functionality focuses on locating computer resources for the specific activity, so it takes less time to complete the operation and consumes less energy. The determination of the best computer resource is based on the usage of ANN correlating with BWO, namely, the ANN-BWO algorithm.

5.3 ANN-BWO scheduler for task scheduling

In the proposed model, the ANN-based task scheduler predicts the scheduling of tasks onto the appropriate physical machine of a specified rack based on the recursive neural network where the weights of it are optimized based on the given Black Widow Optimization algorithm. The entire procedure is given in the form of pseudocode in Algorithm 1.

Algorithm 1: ANN-BWO for task scheduling in Cloud environment

Input: Tasks {T}, # Virtual Machine instances, Execution time calculation function, # input neurons, # Hidden layers, # output neurons

Begin
for each $i = 1\ to\ |neurons|_{IL}$ ***do***
$N_i = T_{j,i} \mid j = 1\ to\ |T|$
end for

// Initialize the weight factors and computation of ANN for every axon

for each $i = 1\ to\ |neurons|_{IL}$ ***do***
$W_i = rand(0,1)$
$WN_i = W_i * N_i$
end for

for each $i = 1\ to\ |neurons|_{HL}$ ***do***
for each j= $1\ to\ |neurons|_{HL}$ ***do***
$W_{i,j} = rand(0,1)$
$W_k N_{i,j} = W_{i,j} * N_i$ *// in sequential order for every subsequent neurons | k = |HL|*
end for
end for

for each $i = 1\ to\ |neurons|_{OL}$ ***do***
$W_i = rand(0,1)$
$W_o N_i = W_i * W_k N$

end for

//Error rate computation

for each $i = 1\ to\ |neurons|_{OL}$ ***do***
$SF_i = Softmax(W_o N_i)$
$ER_i = Abs(AO_i - SF_i)$
end for

// Weight Optimization using BWO

// Input: $\overrightarrow{W}$ *// A weights as a single vector*

While (t<=Max) do

$Y_1, Y_2 = Procreate(W_i, W_j)$
$Y' = Cannibalism\ (Y_1, Y_2)$
$Y'' = Mutate(Y')$
If $(f(Y'') < f(W_i))$ ***then***
$W_i = Y''$
Elseif $(f(Y'') < f(W_j))$ ***then***
$W_j = Y''$
End if
End while

Repeat for all Tasks.

Output: Scheduled Tasks

6 Experimental Results and Discussion

The experimental simulation environment is implemented using CloudSim simulator [30]. **Table 1** shows the simulation environment details. The experimental analysis of the recommended research is evaluated by performing a comparative study on various optimization algorithms such as the Hybrid Particle Swarm Optimization (HPSO), Ant Colony Optimization (ACO), and Intelligent Water Drops (IWD) algorithm. The user submits task requests to the CloudSim by varying the number of tasks in a sequence. The total number of instructions to be performed, file size, length, and more are associated with each cloudlet. Those cloudlets are later sent to a broker to map the VMs you want to use. The host creates VMs that are created in the broker. In a data center broker, the task scheduling algorithm will work. The VMs and data center schedulers can be shared in time or space. We evaluate the experimental results regarding Makespan, Energy consumption, and Cost evaluation with variations in the total number of tasks and total data size.

Table 1
Parameter Settings for CloudSim Simulation.

Type	Parameter	Values
Tasks (Cloudlets)	Number of tasks	100*100
	Length of tasks	[4000-10000] MI
	Input/output file size	[256-1536] MB
	Number of processing element (PE) requirement	1- 4
Virtual machines (VM)	Number of VMs	50
	MIPS or PE	250-2000
	VM memory RAM	512-2048 (MB)
	Bandwidth	500-1000bits
	Type of Manager	Time-shared
Data centers	Number of Data centers	10
	Number of hosts	5-10
	Type of Data centers	Heterogenous
	Type of Manager	Space_shared_time_shared

6.2 Makespan evaluation

It is aimed primarily at reducing the amount of time required, i.e., the whole duration of the scheduled work, where all work has been completed, or the time taken for it from the beginning to the end. The following equation demonstrates the makeup formula.

$$Makespan = \sum_{i=1}^{n} F_{t_i} , \tag{9}$$

where F_{t_i} is the finishing time of the task $t(n)$.

The performance of ANN-BWO is correlated with existing optimization techniques in terms of Makespan (time in seconds), which is presented in **Table 2**.

The performance evaluation of the Makespan (complete task execution time) is discussed in this section. It evaluated 100 to 1000 tasks with different arrival rates such as 20, 40, and 80. The results of the proposed technique ANN-BWO are compared with HPSO, ACO, and IWD. The CloudSim simulation results are shown in **Table 2**. Fig. 4 illustrates the Makespan with an arrival rate of 20. When the total number of tasks is 700, the Makespan of HPSO, ACO, IWD, and proposed ANN-BWO are 497, 474, 468, and 399 seconds, respectively. It is noted that the proposed ANN-BWO outperforms other optimization algorithms regarding Makespan.

Table 2
Comparison of Algorithms with Makespan.

Makespan (S) / Algorithms	Number of Tasks									
	100	200	300	400	500	600	700	800	900	1000
HPSO	227	265	384	436	453	492	497	506	511	516
ACO	236	259	379	434	430	470	474	486	498	511
IWD	215	247	335	387	387	454	468	487	491	524
ANN-BWO	205	220	303	377	372	397	399	448	467	494

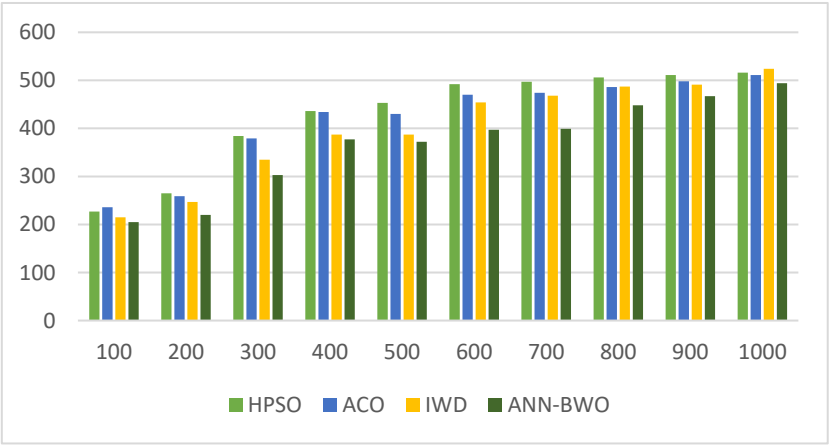

Fig. 4 – *Makespan vs Number of tasks.*

6.3 Energy consumption evaluation

In the **Table 3**, the performance of the proposed ANN-BWO approach is compared with existing techniques in terms of energy for different numbers of tasks is presented.

In addition, the performance analysis of energy consumption with tasks varying between 100-1000 is presented. We can conclude from the experiment results that the higher the number, the larger the energy consumption. **Table 3** and Fig. 5 compare the energy usage of the selected algorithms in this experiment. The findings show that the energy consumption of the ACO algorithm is inadequate when compared with other studied algorithms because half of the scheduling decisions for energy are needed, and half are required to reduce the map. This experiment also demonstrates that our suggested ANN-BWO optimization technique saves energy in the fully loaded as well as moderately loaded cloud environment. The experiment results show that the connections between task load, processing capacity, and energy consumption rates of computer resources can be understood using our trained network.

Table 3
Comparison of algorithms with energy consumption.

Energy Consumption (S) / Algorithms	Number of Tasks									
	100	200	300	400	500	600	700	800	900	1000
HPSO	72	114	132	143	159	166	174	197	228	273
ACO	76	119	144	147	164	174	186	202	233	258
IWD	74	103	129	139	152	159	172	199	226	247
ANN-BWO	70	102	124	136	148	158	167	194	217	239

In addition, the performance analysis of energy consumption with tasks varying between 100-1000 is presented. We can conclude from the experiment results that the higher the number, the larger the energy consumption. **Table 3** and Fig. 5 compare the energy usage of the selected algorithms in this experiment. The findings show that the energy consumption of the ACO algorithm is inadequate when compared with other studied algorithms because half of the scheduling decisions for energy are needed, and half are required to reduce the map. This experiment also demonstrates that our suggested ANN-BWO optimization technique saves energy in the fully loaded as well as moderately loaded cloud environment. The experiment results show that the connections between task load, processing capacity, and energy consumption rates of computer resources can be understood using our trained network.

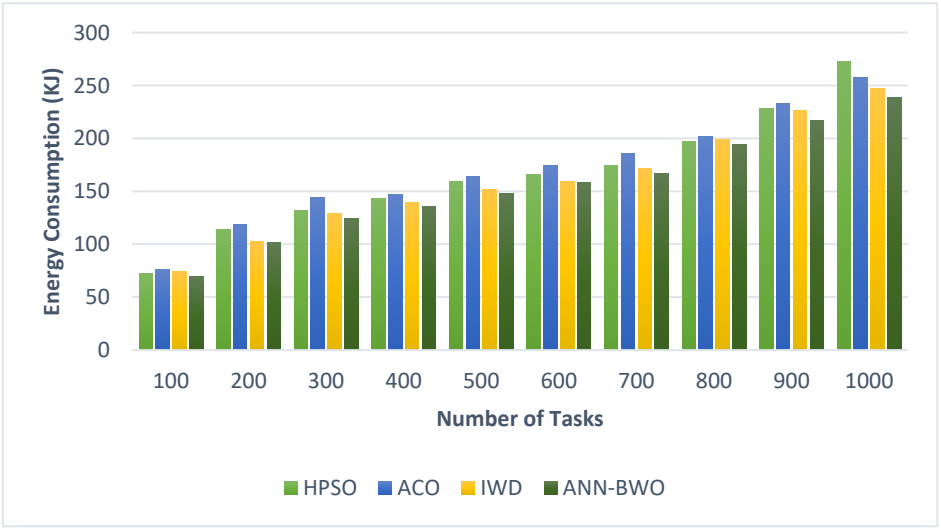

Fig. 5 – *Energy consumption vs Number of tasks.*

6.4 Cost evaluation

The cost evaluation analysis is presented in this section. The execution overhead is another essential indicator that can be used to measure the performance of the task scheduling. The total size of data processed by the workflow ranges from 256 to 1536 MB. By differing the total data size, the variance in the total cost of execution against the distribution of workload on resources is compared for the various algorithms depicted in **Table 4** and Fig. 6, respectively. The total cost of resource computation is assumed in the range 1-1.5$/hr for the experiments. The total cost obtained values of the proposed ANN-BWO are compared with HPSO, ACO, and IWD. From the simulation results, it is evident that the proposed ANN-BWO can complete tasks of varying sizes with minimal cost.

Table 4

Comparison of Algorithms with Cost Evaluation.

Cost Evaluation ($) / Algorithms	Total Data Size (MB)					
	256	512	768	1024	1280	1536
HPSO	45.26	45.72	59.62	64.56	71.74	78.92
ACO	38.32	46.41	62.83	73.69	85.95	98.2
IWD	36.77	45.07	57.89	67.69	78.25	88.81
ANN-BWO	34.86	42.56	51.93	63.24	69.03	84.46

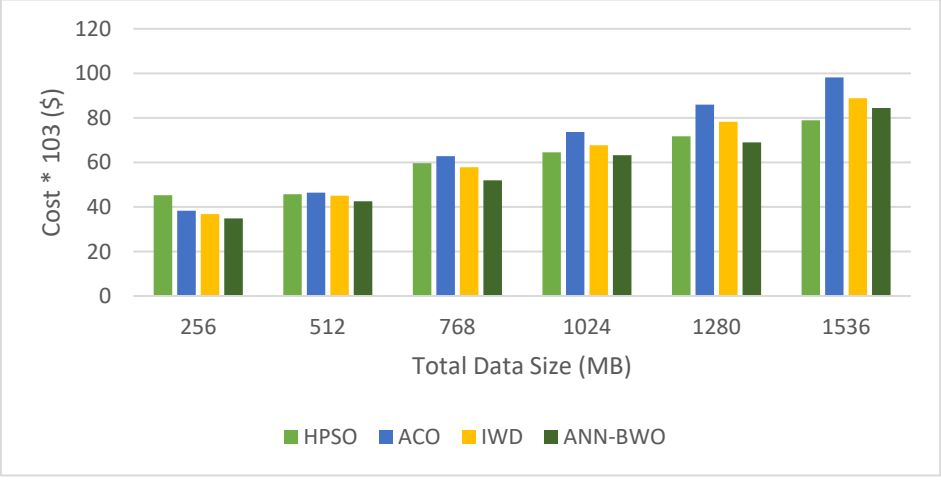

Fig. 6 – *Cost Evaluation vs Number of Tasks.*

7 Conclusion

Cloud computing is a prominent internet paradigm that makes computing a model of service and creates a pool of virtualized computing resources on demand. One of the main issues addressed in Cloud Data Centers is energy-efficient job scheduling. Better Makespan and reduced energy usage indirectly decrease expenses and enhance cloud performance. The volume of carbon emissions from tech infrastructure and cloud data centers is also reduced. For the effective design and deployment of cloud application requirements, adequate knowledge and understanding of performance and energy consumption is significant. In this article, we present ANN-BWO, a novel optimization algorithm for describing energy consumption and performance of cloud-based systems. Using CloudSim, we performed vast experiments to analyze cloud systems energy consumption, performance, and execution overhead with various types and mixes of tasks. We then compared the results of Makespan, energy consumption and execution overhead for cost estimation with different resource allocation strategies. The experimental results reveal that the energy consumption and cost are reduced by approximately 12% and 14%, respectively. We have investigated the energy cost to achieve the best performance, energy, and cost balance in cloud computing systems. For future consideration, we anticipate additionally optimizing the energy consumption and improving the performance in heterogeneous environments like cloud-fog computing, where the fog node's energy consumption and delay description should be examined.

8 References

[1] J. Weinman: The Economics of Pay-per-Use Pricing, IEEE Cloud Computing, Vol. 5, No. 5, September/October 2018, pp. 99 – 107.

[2] L. Wu, S. Kumar Garg, R. Buyya: SLA-Based Admission Control for a Software-as-a-Service Provider in Cloud Computing Environments, Journal of Computer and System Sciences, Vol. 78, No. 5, September 2012, pp. 1280 – 1299.

[3] M. A. Rodriguez, R. Buyya: A Taxonomy and Survey on Scheduling Algorithms for Scientific Workflows in IaaS Cloud Computing Environments, Concurrency and Computation: Practice and Experience, Vol. 29, No. 8, April 2017, p. e4041.

[4] R. Yadav, W. Zhang, K. Li, C. Liu, M. Shafiq, N. Kumar Karn: An Adaptive Heuristic for Managing Energy Consumption and Overloaded Hosts in a Cloud Data Center, Wireless Networks, Vol. 26, No. 3, April 2020, pp. 1905 – 1919.

[5] A. Uchechukwu, K. Li, Y. Shen: Energy Consumption in Cloud Computing Data Centers, International Journal of Cloud Computing and Services Science (IJ-CLOSER), Vol. 3, No. 3, June 2014, pp. 156 – 173.

[6] B. Whitehead, D. Andrews, A. Shah, G. Maidment: Assessing the Environmental Impact of Data Centres Part 1: Background, Energy Use and Metrics, Building and Environment, Vol. 82, December 2014, pp. 151 – 159.

[7] A. Greenberg, J. Hamilton, D. A. Maltz, P. Patel: The Cost of a Cloud: Research Problems in Data Center Networks, ACM SIGCOMM Computer Communication Review, Vol. 39, No. 1, January 2009, pp. 68 – 73.

[8] R. Buyya, A. Beloglazov, J. Abawajy: Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges, arXiv:1006.0308 [cs.DC], June 2010, pp. 1 – 12.

[9] C.- W. Tsai, J. J. P. C. Rodrigues: Metaheuristic Scheduling for Cloud: A Survey, IEEE Systems Journal, Vol. 8, No. 1, March 2014, pp. 279 – 291.

[10] Z. Zhou, J. Chang, Z. Hu, J. Yu, F. Li: A Modified PSO Algorithm for Task Scheduling Optimization in Cloud Computing, Concurrency and Computation: Practice and Experience, Vol. 30, No. 24, December 2018, p. e4970.

[11] S. Eswaran, M. Rajakannu: Unevenness Measurement Using the Support Vector Machine and Dynamic Multiservice Load Balancing with Modified Genetic Algorithm in Cloud-Based Multimedia System, International Journal of Computer Aided Engineering and Technology, Vol. 10, No. 6, October 2018, pp. 732 – 747.

[12] S. Eswaran, M. Rajakannu: Efficient Multimedia Content Storage and Allocation in Multidimensional Cloud Computing Resources, International Journal of Intelligent Systems Technologies and Applications, Vol. 18, No. 1-2, February 2019, pp. 20 – 33.

[13] S. Eswaran, D. Dominic, J. Natarajan, P. B. Honnavalli: Augmented Intelligent Water Drops Optimisation Model for Virtual Machine Placement in Cloud Environment, IET Networks, Vol. 9, No. 5, September 2020, pp. 215 – 222.

[14] M. Sharma, R. Garg: An Artificial Neural Network Based Approach for Energy Efficient Task Scheduling in Cloud Data Centers, Sustainable Computing: Informatics and Systems, Vol. 26, June 2020, p. 100373.

[15] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. M. Abdulhamid, M. J. Usman: Performance Comparison of Heuristic Algorithms for Task Scheduling in IaaS Cloud Computing Environment, PLOS ONE, Vol. 12, No. 5, May 2017, p. e0176321.

[16] M. Cheng, J. Li, S. Nazarian: DRL-Cloud: Deep Reinforcement Learning-Based Resource Provisioning and Task Scheduling for Cloud Service Providers, Proceedings of the 23rd Asia and South Pacific Design Automation Conference (ASP-DAC), Jeju, South Korea, January 2018, pp. 129 – 134.

[17] G. Rjoub, J. Bentahar: Cloud Task Scheduling Based on Swarm Intelligence and Machine Learning, Proceedings of the IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), Prague, Czech Republic, August 2017, pp. 272 – 279.

[18] S. Eswaran, M. Rajakannu: Multiservice Load Balancing with Hybrid Particle Swarm Optimization in Cloud-Based Multimedia Storage System with QoS Provision, Mobile Networks and Applications, Vol. 22, No. 4, August 2017, pp. 760 – 770.

[19] N. Garg, M. S. Goraya: Task Deadline-Aware Energy-Efficient Scheduling Model for a Virtualized Cloud, Arabian Journal for Science and Engineering, Vol. 43, No. 2, February 2018, pp. 829 – 841.

[20] S. Yassa, R. Chelouah, H. Kadima, B. Granado: Multi-Objective Approach for Energy-Aware Workflow Scheduling in Cloud Computing Environments, The Scientific World Journal, Vol. 2013, November 2013, p. ID 350934.

[21] M. Safari, R. Khorsand: PL-DVFS: Combining Power-Aware List-Based Scheduling Algorithm with DVFS Technique for Real-Time Tasks in Cloud Computing, The Journal of Supercomputing, Vol. 74, No. 10, October 2018, pp. 5578 – 5600.

[22] A. Alahmadi, D. Che, M. Khaleel, M.M. Zhu, P. Ghodous: An Innovative Energy-Aware Cloud Task Scheduling Framework, Proceedings of the IEEE 8th International Conference on Cloud Computing, New York, USA, June 2015, pp. 493 – 500.

[23] J.L. Berral, Í. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, J. Torres: Towards Energy-Aware Scheduling in Data Centers Using Machine Learning, Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking, Passau, Germany, April 2010, pp. 215 – 224.

[24] J.L. Berral, R. Gavalda, J. Torres: Adaptive Scheduling on Power-Aware Managed Data-Centers Using Machine Learning, Proceedings of the IEEE/ACM 12th International Conference on Grid Computing, Lyon, France, September 2011, pp. 66 – 73.

[25] B. Tripathy, S. Dash, S.K. Padhy: Dynamic Task Scheduling Using a Directed Neural Network, Journal of Parallel and Distributed Computing, Vol. 75, January 2015, pp. 101 – 106.

[26] A. Agarwal, H. Pirkul, V.S. Jacob: Augmented Neural Networks for Task Scheduling, European Journal of Operational Research, Vol. 151, No. 3, December 2003, pp. 481 – 502.

[27] R. Yang, X. Ouyang, Y. Chen, P. Townend, J. Xu: Intelligent Resource Scheduling at Scale: A Machine Learning Perspective, Proceedings of the IEEE Symposium on Service-Oriented System Engineering (SOSE), Bamberg, Germany, March 2018, pp. 132 – 141.

[28] V. Hayyolalam, A.A.P. Kazem: Black Widow Optimization Algorithm: A Novel Meta-Heuristic Approach for Solving Engineering Optimization Problems, Engineering Applications of Artificial Intelligence, Vol. 87, January 2020, p. 103249.

[29] B. Pourghebleh, V. Hayyolalam: A Comprehensive and Systematic Review of the Load Balancing Mechanisms in the Internet of Things, Cluster Computing, Vol. 23, No. 2, June 2020, pp. 641 – 661.

[30] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya: CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience, Vol. 41, No. 1, January 2011, pp. 23 – 50.